Azure Deployments using Resource Manager templates

In this lab we will create an Azure Resource manager template, we will then separate out our some component resources by linking templates to modularize the resources. We will then modify the main deployment template to call the linked template and updated dependencies, and finally deploy the templates to Azure.

DevOps Course Source

This lab is used in the following courses:

- AZ-400T05: Implementing Application infrastructure - Module 1

Pre-requisites:

- An Azure Subscription

Lab Tasks:

- Task 1: Create Resource Manager template
- Task 2: Create a Linked template for storage resources
- Task 3: Upload Linked Template to Azure Blob Storage and generate SAS token
- Task 4: Modify the main template to call Linked template
- Task 5: Modify main template to update dependencies
- Task 6: Deploy resources to Azure using linked templates

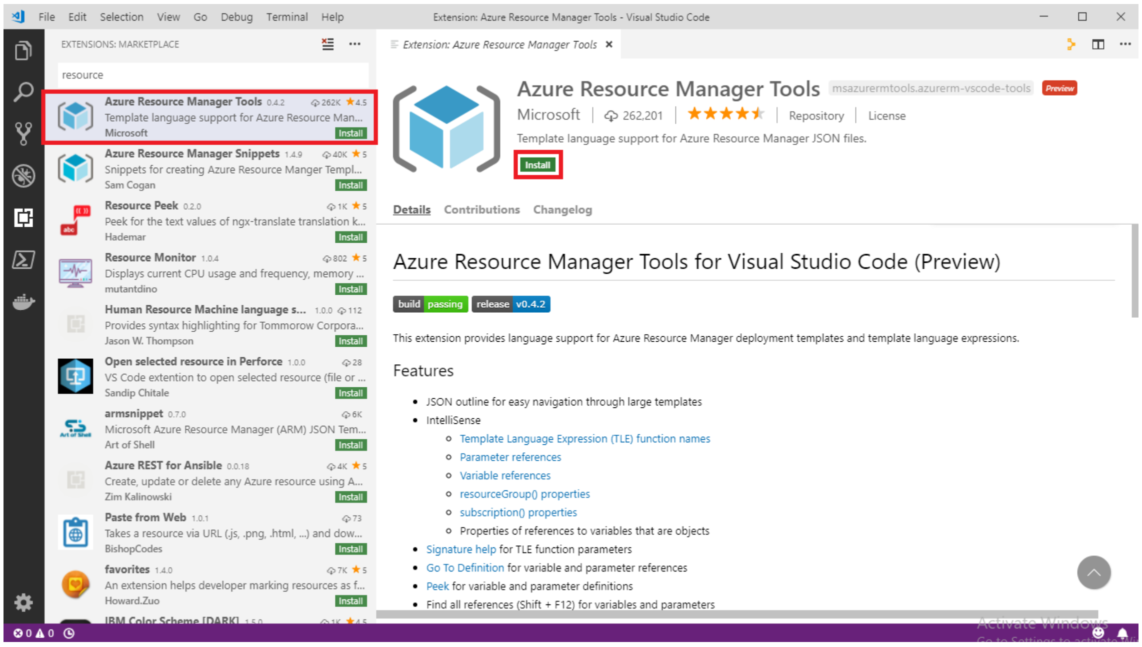
Estimated Lab Time:

- approx. 45 minutes

Task 1: Create Resource Manager template

1. In the task we will use **Visual Studio Code** as our editor, which can be installed from here https://code.visualstudio.com/.

2. In **Visual Studio Code**, go to **File** > **Preferences** > **Extensions** and in the search box type **Azure Resource Manager** tools install the Azure Resource Manager Tools


File Edit Selection View Go Debug Terminal Help
Extension: Azure Resource Manager Tools - Visual Studio Code
EXTENSIONS: MARKETPLACE
resource
Azure Resource Manager Tools
Azure Resource Manager Snippets
Resource Peek
Resource Monitor
Human Resource Machine language s...
Open selected resource in Perforce
armsnippet
Azure REST for Ansible
Paste from Web
favorites
Install
Details Contributions Changelog

Azure Resource Manager Tools msazurermtools.azurerm-vscode-tools Preview

Microsoft | 262,201 | Repository | License

Template language support for Azure Resource Manager JSON files.

Azure Resource Manager Tools for Visual Studio Code (Preview)

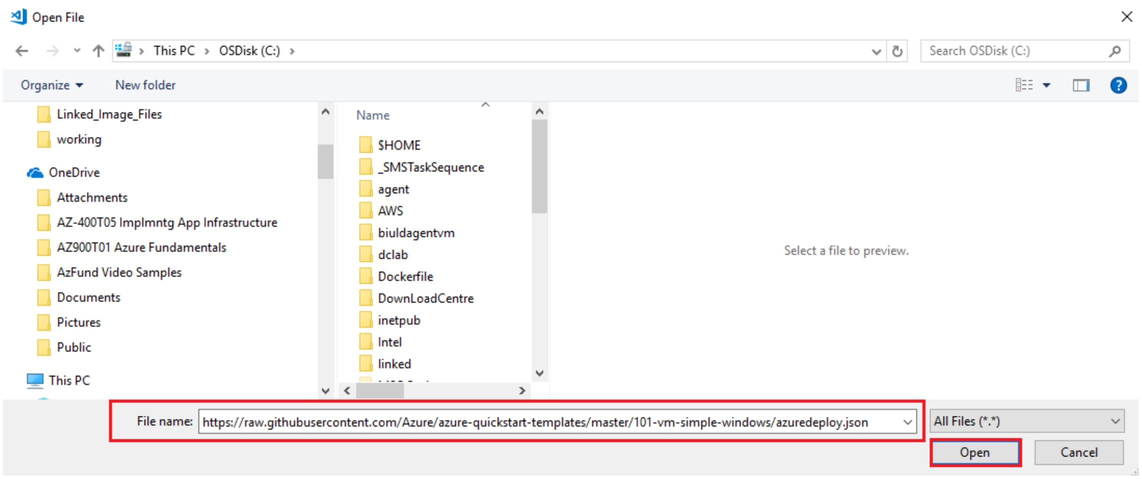
build passing release v0.4.2

This extension provides language support for Azure Resource Manager deployment templates and template language expressions.

Features

- JSON outline for easy navigation through large templates
- IntelliSense
  - Template Language Expression (TLE) function names
  - Parameter references
  - Variable references
  - resourceGroup() properties
  - subscription() properties
  - Properties of references to variables that are objects
- Signature help for TLE function parameters
- Go To Definition for variable and parameter references
- Peek for variable and parameter definitions
- Find all references (Shift + F12) for variables and parameters

3. Open Visual Studio Code and go to **File** > **Open File...** and in the **Open File** dialogue enter the URL https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json and click **Open**

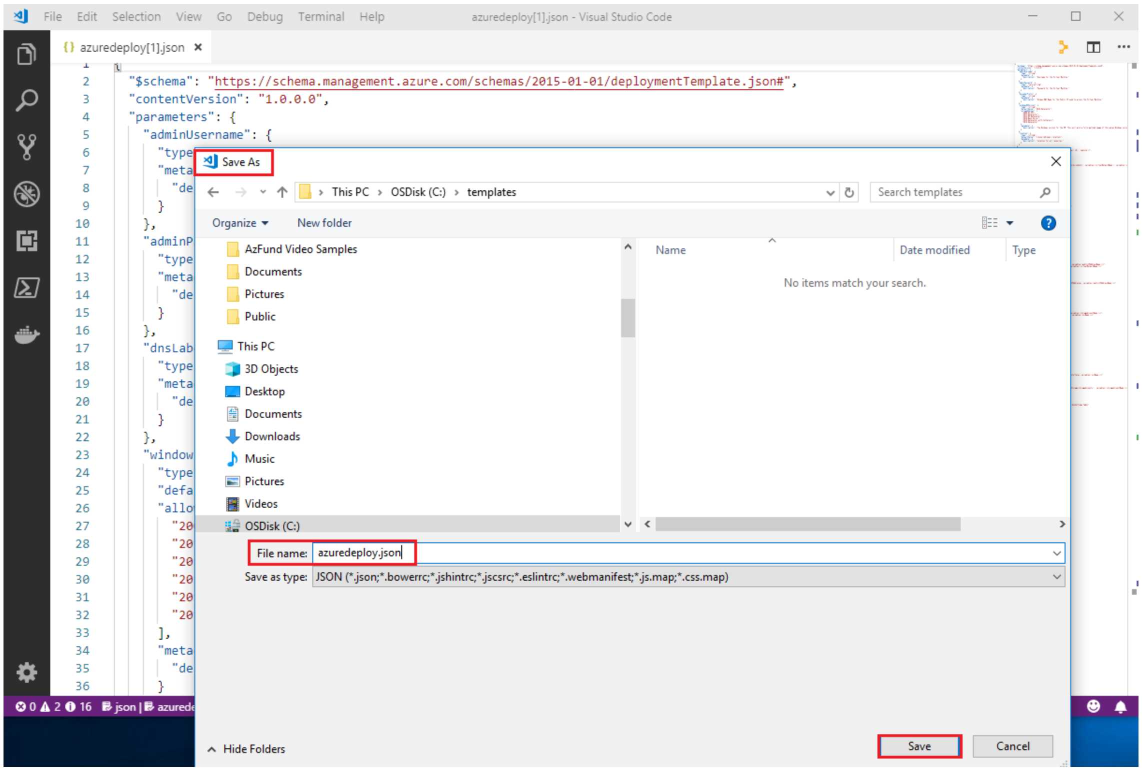


File name: https://raw.githubusercontent.com/Azure/azure-quickstart-templates/master/101-vm-simple-windows/azuredeploy.json

Rather than creating a template from scratch we will use one of the Azure Quickstart Templates. We will use the Deploy a simple Windows template VM. The templates are downloadable the templates from GitHub - 101-vm-simple-windows

4. Create a local folder, you can name it something like C:\Templates, or something like that, this is will be our local working directory.

5. Return to **Visual Studio Code** with our `azuredeploy.json` template open and go to **File** > **Save as** and save the template in our newly created local folder.



```
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "adminUsername": {
```

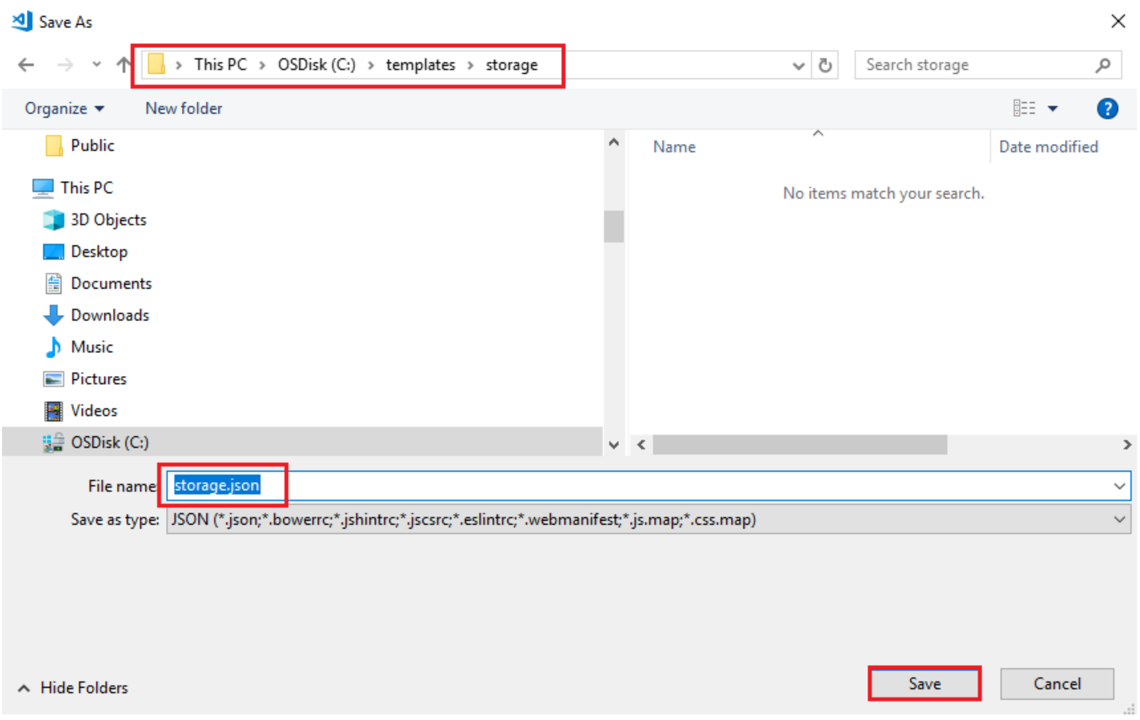
File name: azuredeploy.json

Save as type: JSON (*.json;*.bowerrc;*.jshintrc;*.jscsrc;*.eslintrc;*.webmanifest;*.js.map;*.css.map)

6. Have a look through the resource manager template to get a better understanding of its structure. There are five *resources* defined by the template:

    - Microsoft.Storage/storageAccounts. See the template reference. Microsoft.Storage storageAccounts template reference
    - Microsoft.Network/publicIPAddresses. See the template reference. Microsoft.Network publicIPAddresses template reference
    - Microsoft.Network/virtualNetworks. See the template reference. Microsoft.Network virtualNetworks template reference
    - Microsoft.Network/networkInterfaces. See the template reference. Microsoft.Network networkInterfaces template reference
    - Microsoft.Compute/virtualMachines. See the template reference. Microsoft.Compute virtualMachines template reference

7. In **Visual Studio Code** go to **File** > **Save as...** Create a new folder underneath where the downloaded templates are located called **Storage** so it should be something like `C:\templates\storage`, rename the file `storage.json` or something similar and save it.



File name: storage.json

Save as type: JSON (*.json;*.bowerrc;*.jshintrc;*.jscsrc;*.eslintrc;*.webmanifest;*.js.map;*.css.map)

We now have two identical json files

- C:\templates\azuredeploy.json
- C:\templates\storage\storage.json

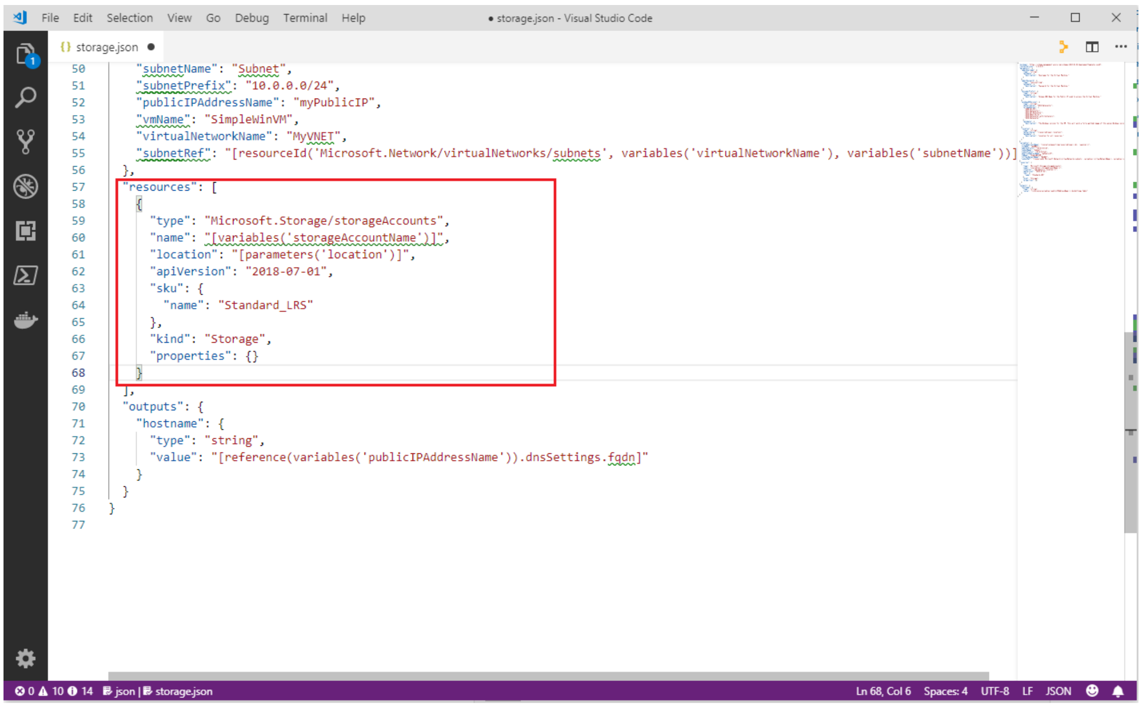
Task 2: Create a Linked template for storage resources

The linked storage template we are creating, `storage.json` will create a storage account. The linked storage template needs to pass a value back to the main template, a`zuredeploy.json`, and this value is defined in the `outputs` element of the linked storage template.

1. Open **Visual Studio Code** and open the `storage.json`. Ensure it is the storage.json that is open and **not** the `azuredeploy.json`.

2. In the `storage.json`, remove all the resource element except the `storageAccounts` resource. It should result in a resource section looking like the below.

```
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[variables('storageAccountName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-07-01",
      "sku": {
        "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {}
      }
```

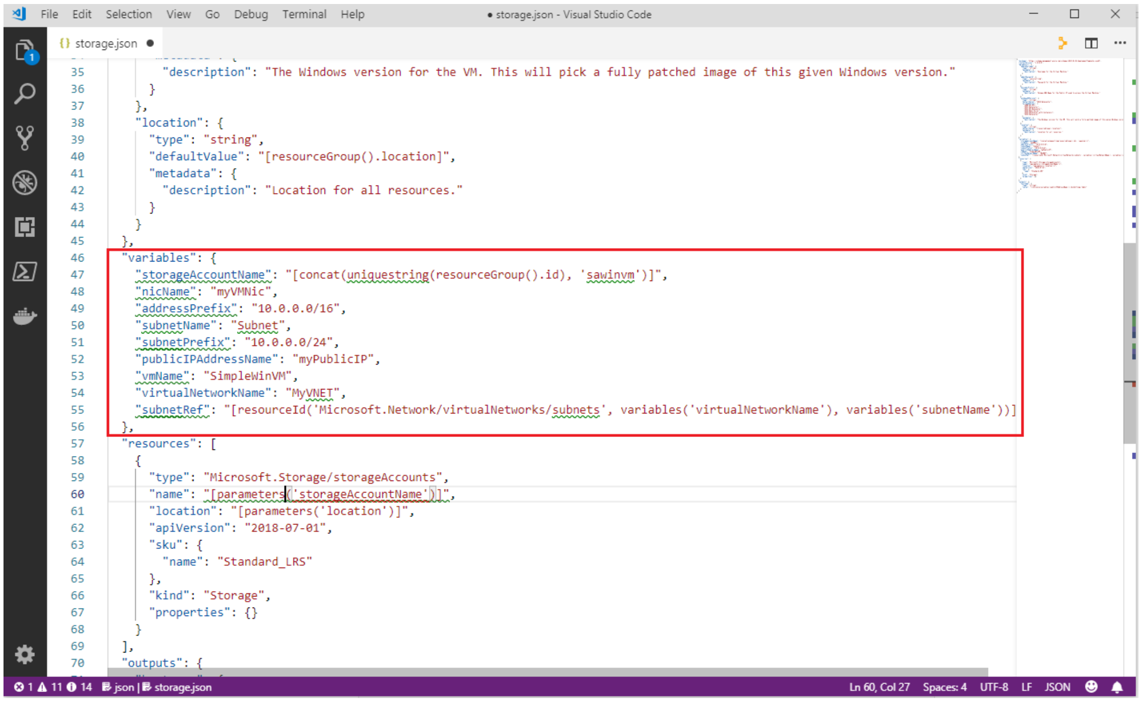


```
      "subnetName": "Subnet",
      "subnetPrefix": "10.0.0.0/24",
      "publicIPAddressName": "myPublicIP",
      "vmName": "SimpleWinVM",
      "virtualNetworkName": "MyVNET",
      "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "name": "[variables('storageAccountName')]",
        "location": "[parameters('location')]",
        "apiVersion": "2018-07-01",
        "sku": {
          "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {}
      }
    ],
    "outputs": {
      "hostname": {
        "type": "string",
        "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]"
      }
    }
  }
```

15. Rename the `name` element of storageAccount from `variables` to `parameters`

```
  "resources": [
{
   "type": "Microsoft.Storage/storageAccounts",
   "name": "[parameters('storageAccountName')]",
   "location": "[parameters('location')]",
   "apiVersion": "2018-07-01",
   "sku": {
     "name": "Standard_LRS"
   },
   "kind": "Storage",
   "properties": {}
}
```

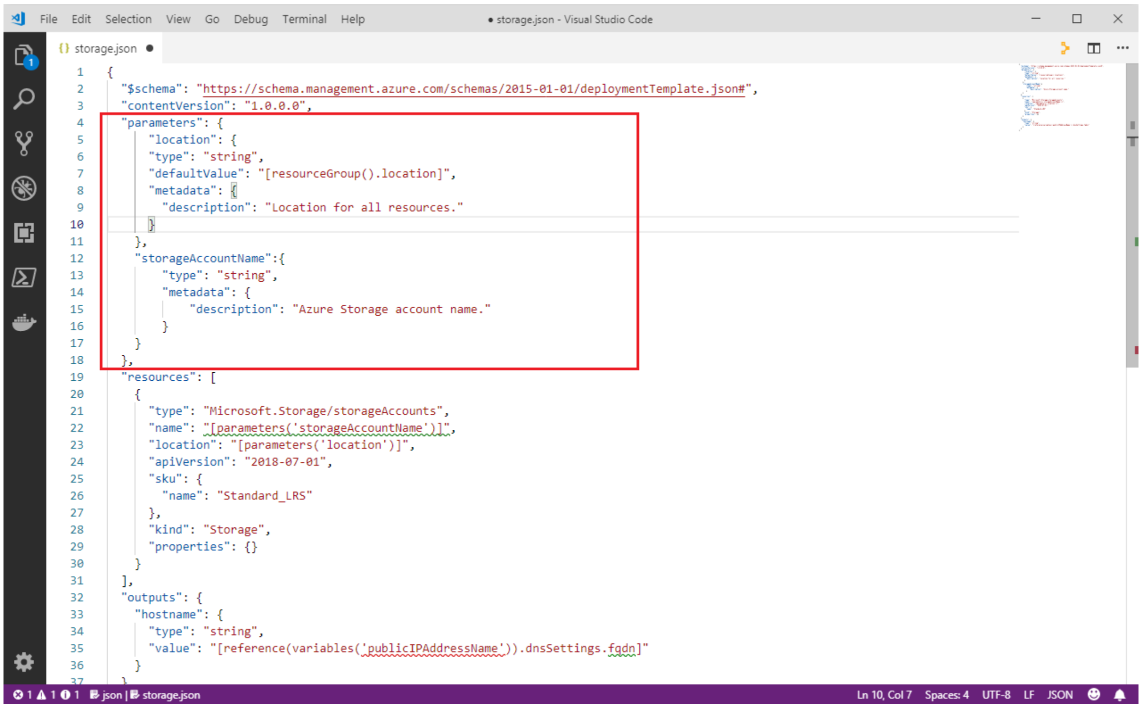
28. Next, remove the `variables` section and all variable definitions, as highlighted below,


File Edit Selection View Go Debug Terminal Help
● storage.json - Visual Studio Code
storage.json
⊗ 1 ▲ 11 ❶ 14 json | storage.json
Ln 60, Col 27 Spaces: 4 UTF-8 LF JSON

```
        "description": "The Windows version for the VM. This will pick a fully patched image of this given Windows version."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "variables": {
    "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'sawinvm')]",
    "nicName": "myVMNic",
    "addressPrefix": "10.0.0.0/16",
    "subnetName": "Subnet",
    "subnetPrefix": "10.0.0.0/24",
    "publicIPAddressName": "myPublicIP",
    "vmName": "SimpleWinVM",
    "virtualNetworkName": "MyVNET",
    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]"
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-07-01",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    }
  ],
  "outputs": {
```

29. Next, remove all `parameter` values except **location** and add the following paramter code. It should end up looking as in the screenshot below.

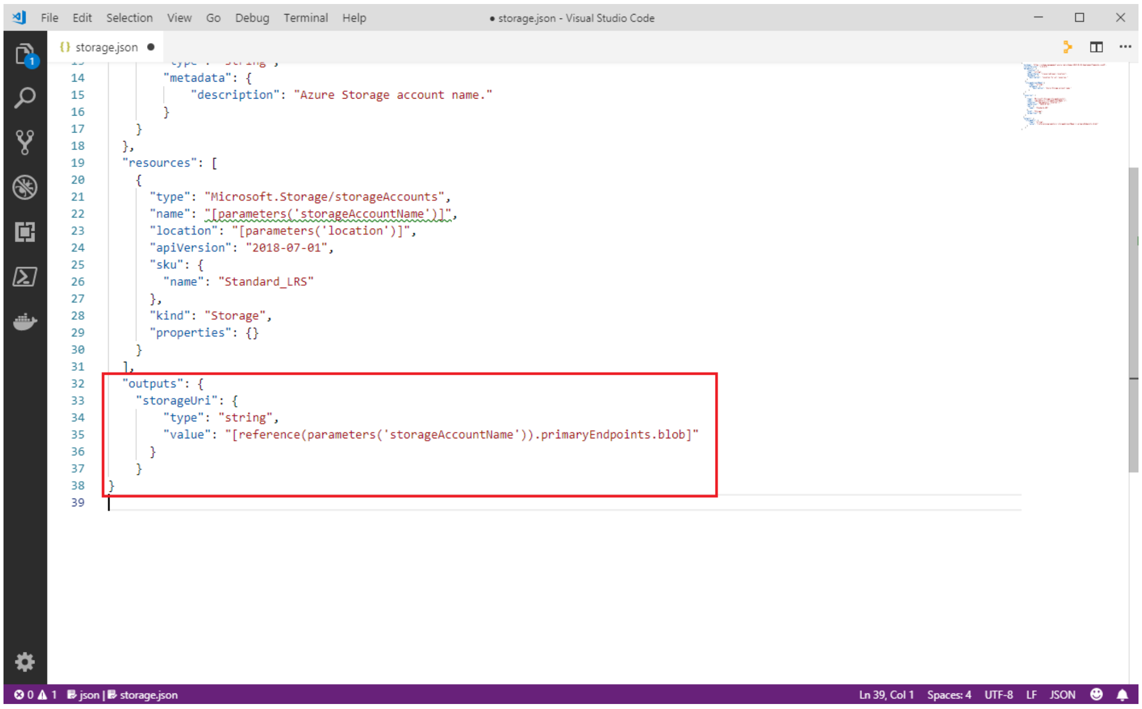
```
"storageAccountName":{
"type": "string",
"metadata": {
    "description": "Azure Storage account name."
}
},
```


File Edit Selection View Go Debug Terminal Help
● storage.json - Visual Studio Code
storage.json
⊗ 1 ▲ 1 ❶ 1 json | storage.json
Ln 10, Col 7 Spaces: 4 UTF-8 LF JSON

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "location": {
    "type": "string",
    "defaultValue": "[resourceGroup().location]",
    "metadata": {
      "description": "Location for all resources."
    }
  },
  "storageAccountName":{
    "type": "string",
    "metadata": {
        "description": "Azure Storage account name."
    }
  }
},
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('storageAccountName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-07-01",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    }
  ],
  "outputs": {
    "hostname": {
      "type": "string",
      "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]"
    }
```

36. Next, update the `output` section to define a `storageURI` output value. The `storageUri` value is required by the virtual machine resource definition in the main template. You pass the value back to the main template as an output value. Modify the output so it looks like the below.

```
"outputs": {
    "storageUri": {
        "type": "string",
        "value": "[reference(parameters('storageAccountName')).primaryEndpoints.blob]"
      }
}
```



File Edit Selection View Go Debug Terminal Help • storage.json - Visual Studio Code

```
        "metadata": {
            "description": "Azure Storage account name."
        }
      }
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "name": "[parameters('storageAccountName')]",
        "location": "[parameters('location')]",
        "apiVersion": "2018-07-01",
        "sku": {
          "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {}
      }
    ],
    "outputs": {
      "storageUri": {
        "type": "string",
        "value": "[reference(parameters('storageAccountName')).primaryEndpoints.blob]"
      }
    }
}
```

⊗ 0 ⚠ 1 json | storage.json Ln 39, Col 1 Spaces: 4 UTF-8 LF JSON

43. Save the `storage.json` template. The linked storage template should now look like the below,

```
{
  "$schema": "https://schema.management.azure.com/schemas/2015-01-01/deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "storageAccountName":{
      "type": "string",
      "metadata": {
        "description": "Azure Storage account name."
      }
    },
    "location": {
      "type": "string",
      "defaultValue": "[resourceGroup().location]",
      "metadata": {
        "description": "Location for all resources."
      }
    }
  },
  "resources": [
    {
      "type": "Microsoft.Storage/storageAccounts",
      "name": "[parameters('storageAccountName')]",
      "apiVersion": "2016-01-01",
      "location": "[parameters('location')]",
      "sku": {
        "name": "Standard_LRS"
      },
      "kind": "Storage",
      "properties": {}
    }
  ],
  "outputs": {
      "storageUri": {
          "type": "string",
          "value": "[reference(parameters('storageAccountName')).primaryEndpoints.blob]"
        }
  }
}

```

Task 3: Upload Linked Template to Azure Blob Storage and generate SAS token

When linking to a template, the Azure Resource Manager service must be able to access it. As such You **cannot** specify a local file or a file that is only available on your local network. You can only provide a URI value that includes either `http` or `https`.

In order to achieve this we will upload our linked storage template, `storage.json`, to blob storage in Azure. Then we will generate a url that we can then use to access it.

We will perform these steps in the Azure CLI, and we would recommend to do so for ease of use. Azure Cloud Shell, has the latest **Az** powershell module installed ready for use. however you could manually create a blob container via the Azure Portal, upload thw file and generate a URL, or run the powershell commands locally, however ensure you have the latest powershell modules installed locally if running from a local environment.

1. Open the Azure Cloud Shell via http://shell.azure.com
2. Switch the Azure Cloud Shell to a **PowerShell** command line.
3. The below is a PowerShell script, that will create a blob storage container, upload our template file and generate s URL with a SAS token that we can call in our main template. Have a read through the below to understand what it is doing. When youi are ready copy and paste the below into the **Azure Cloud Shell** and enter values as prompted. i.e.

   Note: Ensure you note the value that are output at the end of the script, as they will be required later in the lab.

```
$projectNamePrefix = Read-Host -Prompt "Enter a project name:"   # This name is used to generate names for Azure resources, such as storage account name.

$location = Read-Host -Prompt "Enter a location (i.e. centralus)" # use your nearest datacenter


$resourceGroupName = $projectNamePrefix + "rg"

$storageAccountName = $projectNamePrefix + "stracc"

$containerName = "linktempblobcntr" # The name of the Blob container to be created.


$linkedTemplateURL = "https://raw.githubusercontent.com/Microsoft/PartsUnlimited/master/Labfiles/AZ-400T05_Implementing_Application_Infrastructure/M01/storage.json" # A completed linked template used in this lab.

$fileName = "storage.json" # A file name used for downloading and uploading the linked template.


# Download the lab linked template

Invoke-WebRequest -Uri $linkedTemplateURL -OutFile "$home/$fileName" # This generates a copy of the storage.json template in your Azure Cloud Shell session
```

```
# Create a resource group
New-AzResourceGroup -Name $resourceGroupName -Location $location # creates a new resource group into which we create a storage account, then a blob container

# Create a storage account
$storageAccount = New-AzStorageAccount `
    -ResourceGroupName $resourceGroupName `
    -Name $storageAccountName `
    -Location $location `
    -SkuName "Standard_LRS"

$context = $storageAccount.Context

# Create a container
New-AzureStorageContainer -Name $containerName -Context $context


# Upload the linked template
Set-AzureStorageBlobContent `
    -Container $containerName `
    -File "$home/$fileName" `
    -Blob $fileName `
    -Context $context

# Generate a SAS token. We set an expiry time of 24 hours, but you could have shorter values for increased security.
$templateURI = New-AzureStorageBlobSASToken `
    -Context $context `
    -Container $containerName `
    -Blob $fileName `
    -Permission r `
    -ExpiryTime (Get-Date).AddHours(24.0) `
    -FullUri

echo "You need the following values later in the tutorial:"
echo "Resource Group Name: $resourceGroupName"
echo "Linked template URI with SAS token: $templateURI"

echo "finished"
```

4. Note the output values of the below, they should look something like the below.

    - **$resourcegroupName**: lnkedtempprojrg
    - **$templateURI**: https://lnkedtempprojstracc.blob.core.windows.net/linktempblobcntr/storage.json?sv=2018-03-28&sr=b&sig=B4hDLt9rFaWHZXToJlMwMjejAQGT7x0INdDR9bHBQnI%3D&se=2019-02-05T10%3A09%3A48Z&sp=r

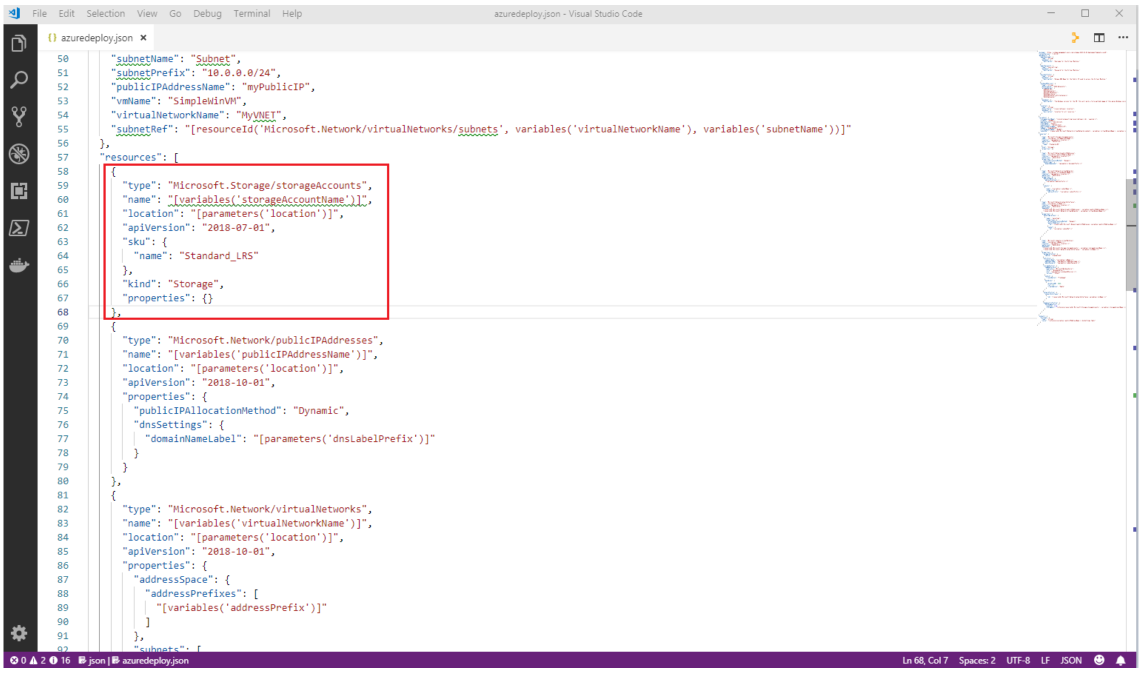
    For scenarios requiring more security we could configure the main template to generate a SAS token when the template is being run, and give the SAS token expiry a smaller window to make it more secure. You can see the Deploy private Resource Manager template with SAS token and Azure PowerShell page for more details on how to do this.

Task 4: Modify the main template to call Linked template

To account for the changes we made to the templates structure by modularizing all the storage elements, we now need to modify the main template to call the new storage resource definition.

1. In **Visual Studio Code** open the main template `azuredeploy.json`.
2. In the `resource` section remove the `storage` resource element

```
{
  "type": "Microsoft.Storage/storageAccounts",
  "name": "[variables('storageAccountName')]",
  "location": "[parameters('location')]",
  "apiVersion": "2018-07-01",
  "sku": {
    "name": "Standard_LRS"
  },
  "kind": "Storage",
  "properties": {}
},
```

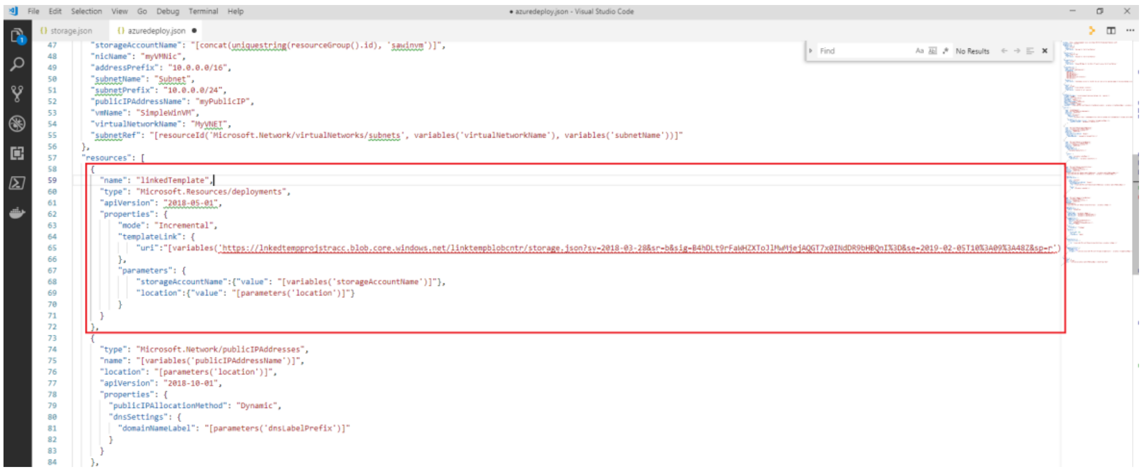

File Edit Selection View Go Debug Terminal Help
azuredeploy.json - Visual Studio Code
azuredeploy.json
0 2 16 json | azuredeploy.json
Ln 68, Col 7 Spaces: 2 UTF-8 LF JSON

```
      "subnetName": "Subnet",
      "subnetPrefix": "10.0.0.0/24",
      "publicIPAddressName": "myPublicIP",
      "vmName": "SimpleWinVM",
      "virtualNetworkName": "MyVNET",
      "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]"
    },
    "resources": [
      {
        "type": "Microsoft.Storage/storageAccounts",
        "name": "[variables('storageAccountName')]",
        "location": "[parameters('location')]",
        "apiVersion": "2018-07-01",
        "sku": {
          "name": "Standard_LRS"
        },
        "kind": "Storage",
        "properties": {}
      },
      {
        "type": "Microsoft.Network/publicIPAddresses",
        "name": "[variables('publicIPAddressName')]",
        "location": "[parameters('location')]",
        "apiVersion": "2018-10-01",
        "properties": {
          "publicIPAllocationMethod": "Dynamic",
          "dnsSettings": {
            "domainNameLabel": "[parameters('dnsLabelPrefix')]"
          }
        }
      },
      {
        "type": "Microsoft.Network/virtualNetworks",
        "name": "[variables('virtualNetworkName')]",
        "location": "[parameters('location')]",
        "apiVersion": "2018-10-01",
        "properties": {
          "addressSpace": {
            "addressPrefixes": [
              "[variables('addressPrefix')]"
            ]
          },
          "subnets": [
```

14. Next, add the following code to the `resources` section where the storage element was that you just deleted.

    Note: Ensure to paste in the `$templateURI` value that you obtained when you uploaded the linked `storage.json` to blob storage earlier.

```
{
  "name": "linkedTemplate",
  "type": "Microsoft.Resources/deployments",
  "apiVersion": "2018-05-01",
  "properties": {
      "mode": "Incremental",
      "templateLink": {
          "uri":"< enter the Template URI value you obtained earlier containing a SAS Token >"
      },
      "parameters": {
          "storageAccountName":{"value": "[variables('storageAccountName')]"},
          "location":{"value": "[parameters('location')]"}
      }
  }
},
```


File Edit Selection View Go Debug Terminal Help
● azuredeploy.json - Visual Studio Code
storage.json
azuredeploy.json
Find
No Results

```
    "storageAccountName": "[concat(uniquestring(resourceGroup().id), 'sawinvm')]",
    "nicName": "myVMNic",
    "addressPrefix": "10.0.0.0/16",
    "subnetName": "Subnet",
    "subnetPrefix": "10.0.0.0/24",
    "publicIPAddressName": "myPublicIP",
    "vmName": "SimpleWinVM",
    "virtualNetworkName": "MyVNET",
    "subnetRef": "[resourceId('Microsoft.Network/virtualNetworks/subnets', variables('virtualNetworkName'), variables('subnetName'))]"
  },
  "resources": [
    {
      "name": "linkedTemplate",
      "type": "Microsoft.Resources/deployments",
      "apiVersion": "2018-05-01",
      "properties": {
        "mode": "Incremental",
        "templateLink": {
          "uri":"[variables('https://lnkedtempprojstracc.blob.core.windows.net/linktempblobcntr/storage.json?sv=2018-03-28&sr=b&sig=B4hDLt9rFaWHZXToJlMwMjejAQGT7x0INdDR9bHBQnI%3D&se=2019-02-05T10%3A09%3A48Z&sp=r')
        },
        "parameters": {
          "storageAccountName":{"value": "[variables('storageAccountName')]"},
          "location":{"value": "[parameters('location')]"}
        }
      }
    },
    {
      "type": "Microsoft.Network/publicIPAddresses",
      "name": "[variables('publicIPAddressName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-10-01",
      "properties": {
        "publicIPAllocationMethod": "Dynamic",
        "dnsSettings": {
          "domainNameLabel": "[parameters('dnsLabelPrefix')]"
        }
      }
    },
```

Note the following details from this section above:

- A `Microsoft.Resources/deployments` resource in the main template is used to link to another template.
- The `deployments` resource has a name called `linkedTemplate`. This name is used for configuring dependency.
- You can only use `Incremental` deployment mode when calling linked templates.
- `templateLink/uri` contains the linked template URI. Update the value to the URI you get when you upload the linked template (the one with a SAS token).
- Use `parameters` to pass values from the main template to the linked template.

15. Save the template.

Task 5: Modify main template to update dependencies

1. Next, because the storage account is defined in the linked storage template now, we must update the following two elements of the `Microsoft.Compute/virtualMachines` resource definition. In the resource section in the virtual machines element, update the `dependOn` element to the below:

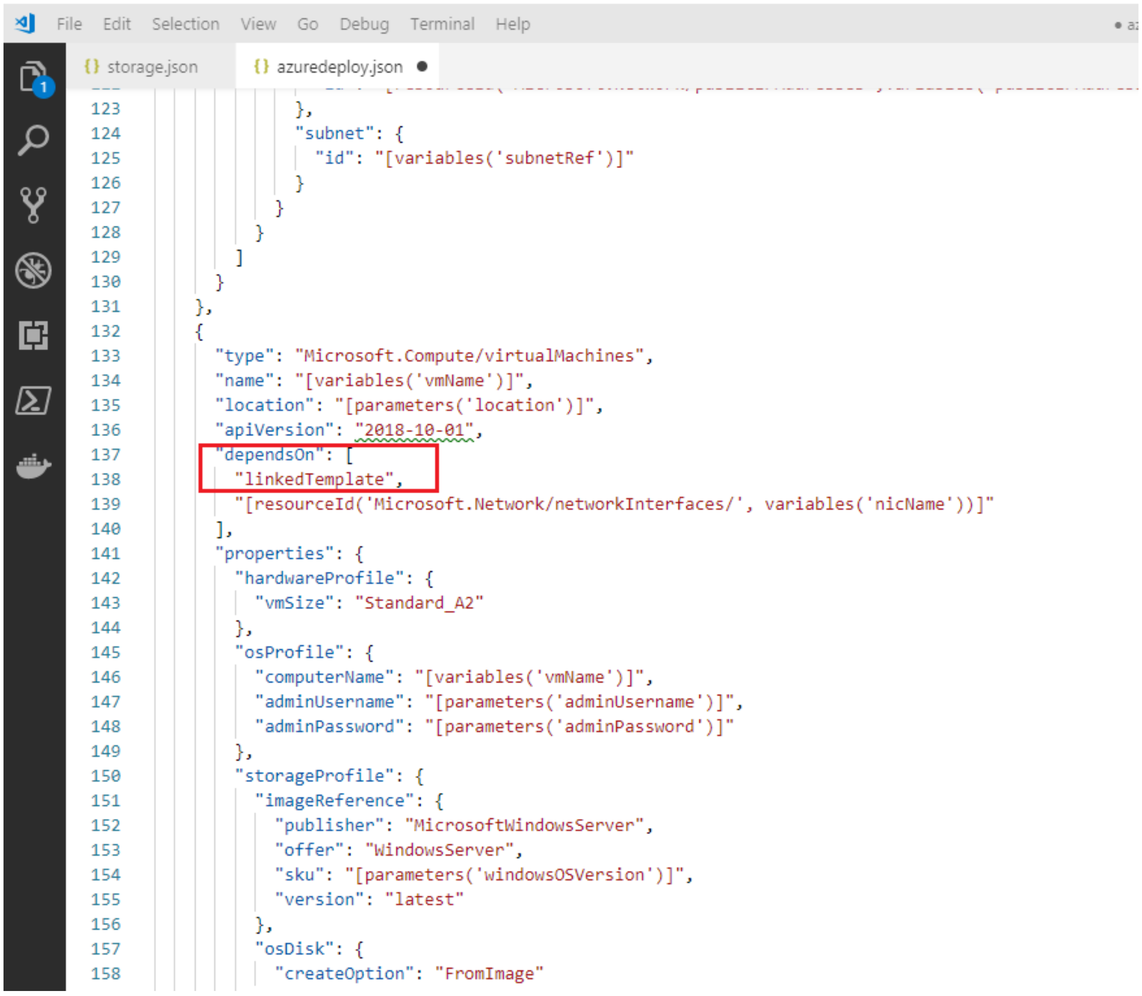
   from

   ```
          "dependsOn": [
            "[resourceId('Microsoft.Storage/storageAccounts/', variables('storageAccountName'))]",
            "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
   ```

   to

   ```
          "dependsOn": [
            "linkedTemplate",
   ```

```
          "[resourceId('Microsoft.Network/networkInterfaces/', variables(
'nicName'))]"
```



```
              },
              "subnet": {
                "id": "[variables('subnetRef')]"
              }
            }
          }
        ]
      }
    },
    {
      "type": "Microsoft.Compute/virtualMachines",
      "name": "[variables('vmName')]",
      "location": "[parameters('location')]",
      "apiVersion": "2018-10-01",
      "dependsOn": [
        "linkedTemplate",
        "[resourceId('Microsoft.Network/networkInterfaces/', variables('nicName'))]"
      ],
      "properties": {
        "hardwareProfile": {
          "vmSize": "Standard_A2"
        },
        "osProfile": {
          "computerName": "[variables('vmName')]",
          "adminUsername": "[parameters('adminUsername')]",
          "adminPassword": "[parameters('adminPassword')]"
        },
        "storageProfile": {
          "imageReference": {
            "publisher": "MicrosoftWindowsServer",
            "offer": "WindowsServer",
            "sku": "[parameters('windowsOSVersion')]",
            "version": "latest"
          },
          "osDisk": {
            "createOption": "FromImage"
```

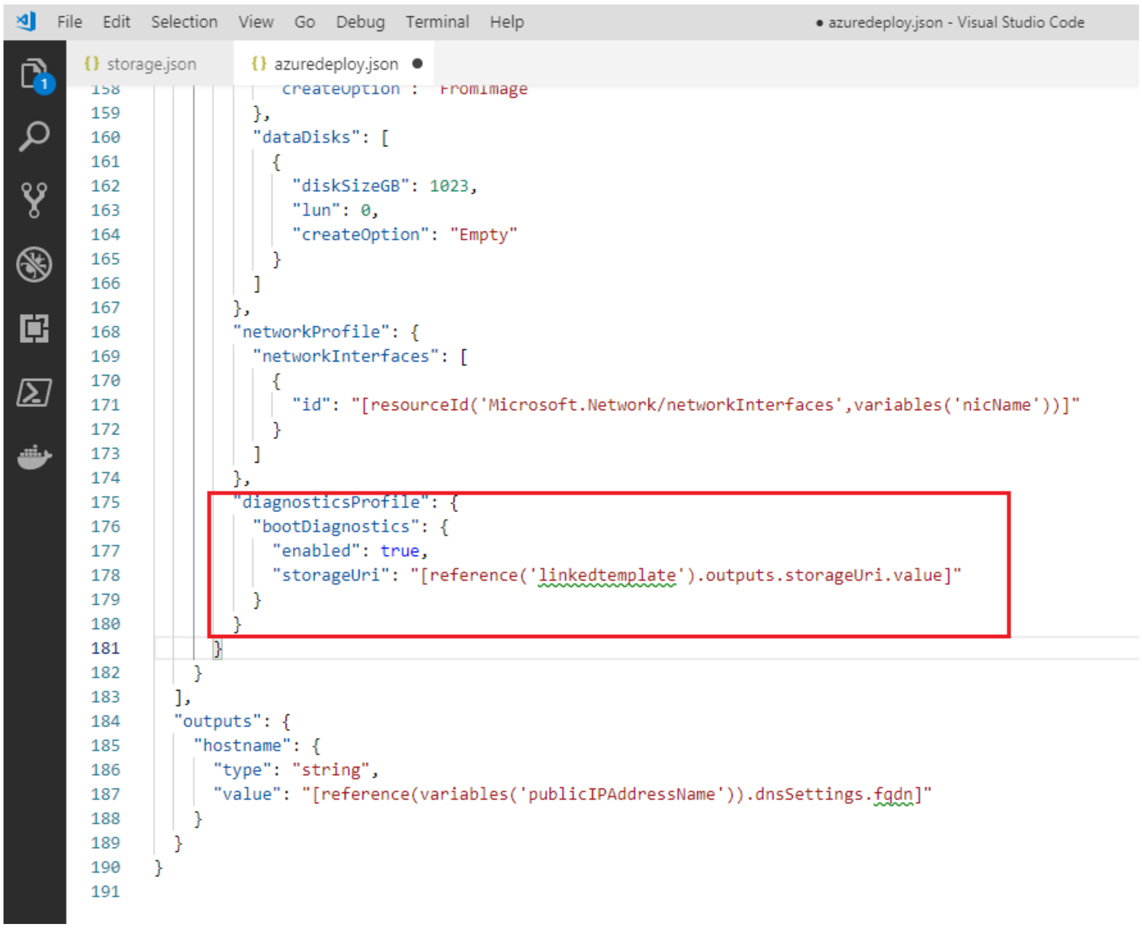
2. Still in the `resources` section under the `Microsoft.Compute/virtualMachines` element, reconfigure the `properties/diagnosticsProfile/bootDiagnostics/storageUri` element to the output value you defined in the linked storage template as below. This value is required by the main template

   from

```
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "[reference(resourceId('Microsoft.Storage/stor
ageAccounts/', variables('storageAccountName'))).primaryEndpoints.blob]"
          }
```

   to

```
        "diagnosticsProfile": {
          "bootDiagnostics": {
            "enabled": true,
            "storageUri": "[reference('linkedtemplate').outputs.storageUri.value]"
          }
```


File Edit Selection View Go Debug Terminal Help
● azuredeploy.json - Visual Studio Code
storage.json
azuredeploy.json ●

```
              "createOption": "FromImage"
            },
            "dataDisks": [
              {
                "diskSizeGB": 1023,
                "lun": 0,
                "createOption": "Empty"
              }
            ]
          },
          "networkProfile": {
            "networkInterfaces": [
              {
                "id": "[resourceId('Microsoft.Network/networkInterfaces',variables('nicName'))]"
              }
            ]
          },
          "diagnosticsProfile": {
            "bootDiagnostics": {
              "enabled": true,
              "storageUri": "[reference('linkedtemplate').outputs.storageUri.value]"
            }
          }
        }
      }
    ],
    "outputs": {
      "hostname": {
        "type": "string",
        "value": "[reference(variables('publicIPAddressName')).dnsSettings.fqdn]"
      }
    }
  }
```

3. Save the updated main deployment template.

Task 6: Deployed resources to Azure using linked templates

You can deploy templates using a variety of methods, including directly from the Azure Portal, or using Azure CLI or Powershell, locally, or via the Azure Cloud Shell. We will will use Azure CLI run from a local command line.

If we wanted to do the deployment using in the Azure Cloud Shell, we would just need to ensure the main deployment template, `azuredeploy.json`, was accessible from the Azure Cloud Shell. We could do that by uploading the `azuredeploy.json` to blob storage and obtaining the file URL. Then when running an Azure CLI command we would use a `--template-uri` parameter rather than a `--template-file` parameter that we use below.

1. Open a local command prompt

2. Sign in to Azure by running the below command. If the CLI can open your default browser, it will do so and load a sign-in page. Otherwise, you need to open a browser page and follow the instructions on the command line to enter an authorization code after navigating to `https://aka.ms/devicelogin` in your browser.

```
3.  az login
```

4. Sign in with your Azure credentials and ensure you are in the same subscription context as the resource group you created earlier. You cna check this using the command

```
5.  az account list
```

   and if you need ot change the account context you can use the command

```
 az account set --subscription < your subscription name or ID>
```

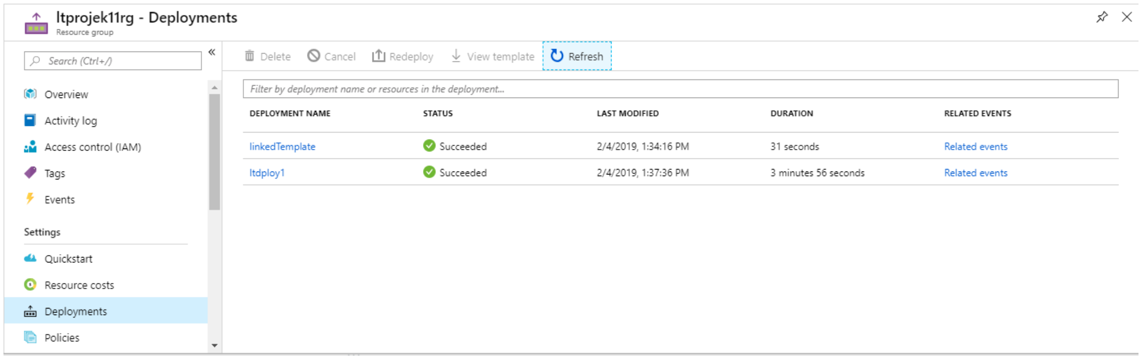
6. Deploy the template by running the command

```
7.  az group deployment create --name < name for the deployment > --resourc
e-group < resource group you created earlier when running PowerShell scr
ipt >  --template-file < path to the main template file i.e. C:\Template
s\azuredeploy.json >
```

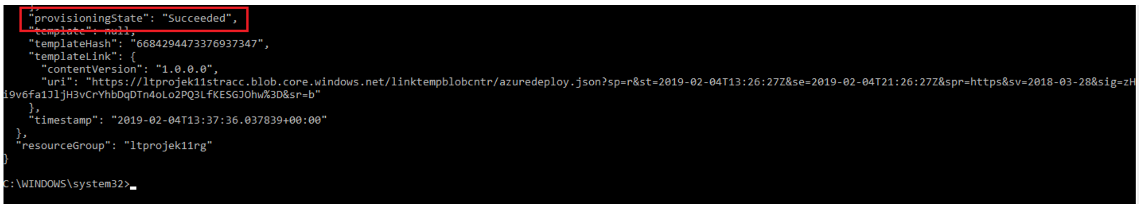
   If you receive errors when running the above command to deploy the templates you could check the following:

   - If you have multiple Azure subscriptions ensure you have set the subscription context to the correct one where the resource group is deployed.
   - Ensure the files are accessible via the URLs you have provided

8. Either via the Azure cli command output or the Azure Portal verify the virtual machine and storage resources were all created successfully


ltprojek11rg - Deployments
Resource group

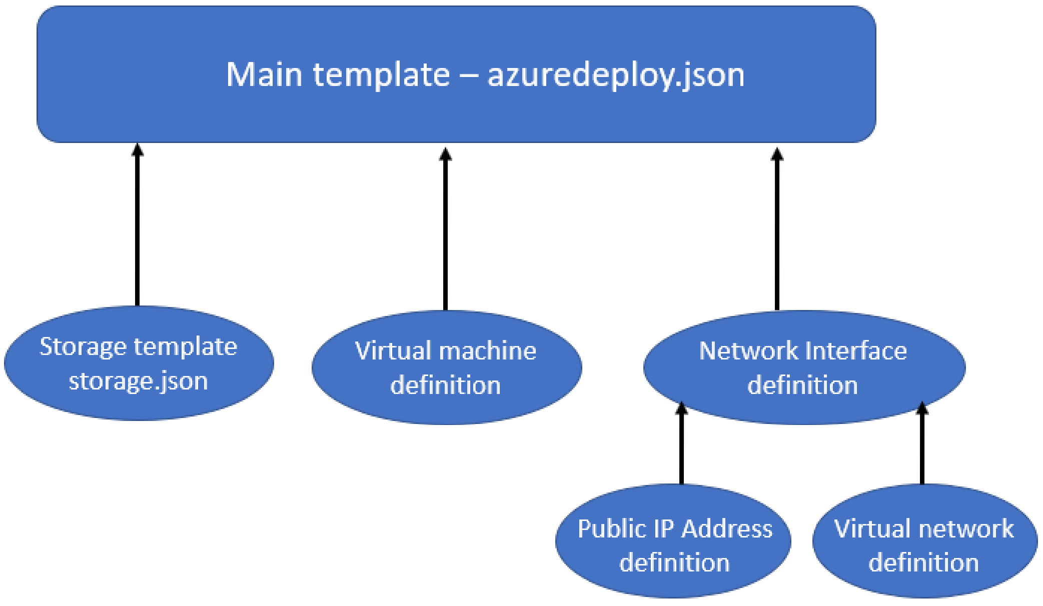
DEPLOYMENT NAME	STATUS	LAST MODIFIED	DURATION	RELATED EVENTS
linkedTemplate	✓ Succeeded	2/4/2019, 1:34:16 PM	31 seconds	Related events
ltdploy1	✓ Succeeded	2/4/2019, 1:37:36 PM	3 minutes 56 seconds	Related events



```
    "provisioningState": "Succeeded",
    "template": null,
    "templateHash": "6684294473376937347",
    "templateLink": {
      "contentVersion": "1.0.0.0",
      "uri": "https://ltprojek11stracc.blob.core.windows.net/linktempblobcntr/azuredeploy.json?sp=r&st=2019-02-04T13:26:27Z&se=2019-02-04T21:26:27Z&spr=https&sv=2018-03-28&sig=zH
i9v6fa1JljH3vCrYhbDqDTn4oLo2PQ3LfKESGJOhw%3D&sr=b"
    },
    "timestamp": "2019-02-04T13:37:36.037839+00:00"
  },
  "resourceGroup": "ltprojek11rg"
}

C:\WINDOWS\system32>
```

9. As a next step we could now proceed with modularizing the remaining resource definitions in our deployment template, such as our network and virtual machine resource definitions. A way to approach this would be to modularize based on dependencies. A graphical representing of the dependencies in the template we use here is shown below.


Main template – azuredeploy.json
Storage template storage.json
Virtual machine definition
Network Interface definition
Public IP Address definition
Virtual network definition

Note: If you do not intend using the deployed resources you should delete them now to avoid incurring costs related ot them. You can do so by deleting the resource groups.

Summary

In this lab you have:

- Created Resource Manager template by re-using a Quickstart template
- Created a Linked template for storage resources by defining just storage definitions
- Uploaded the Linked storage template to Azure Blob Storage and generates SAS token to be called by Main deployment template.
- Modified the main template to call Linked template
- Modified main template to update dependencies
- Deployed resources to Azure using linked templates